AMENDMENTS TO THE CLAIMS

1. (Currently amended) ~~A method of generating an intermediate representation during translation of subject code into target code, comprising the computer-implemented steps of:~~

~~decoding a plurality of instructions in the subject code;~~

~~generating an intermediate representation of the decoded instructions including providing a plurality of nodes in the intermediate representation as abstract representations of the expressions, calculations, and operations performed by the instructions of the subject code selected from a plurality of possible types of nodes including at least base nodes and complex nodes, wherein the base nodes represent the most basic semantics of the subject code such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics, and wherein the complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations; and~~

~~determining at least one type of node out of the plurality of possible types of nodes to generate in the intermediate representation for each respective instruction in the decoded subject code.~~ A method of converting program code of a subject computing architecture into target code executed on a target computing system, comprising the steps of:

decoding a plurality of instructions in the program code;

determining which type of nodes to generate in an intermediate representation for each of the decoded instructions in the program code, including determining that one or more of the decoded instructions require base nodes having basic RISC-like functionality which provides an expanded representation of semantics of the decoded instructions, and one or more of the decoded instructions require complex nodes having complex CISC-like functionality which provides a compact representation of semantics of the decoded instructions;

generating the intermediate representation of the decoded instructions using the determinced types of nodes, including generating the base nodes and the complex nodes in the intermediate representation from the respective decoded instructions;

generating the target code from the intermediate representation; and

executing the target code on the target computing system.

2. (Canceled)

3. (Canceled)

4. (Currently amended) The method of claim 1, wherein the base nodes are generic across a plurality of possible subject computing architectures.

5. (Canceled)

6. (Currently amended) The method of claim 1, wherein the determining step includes that the program code includes ~~the complex nodes represent~~ immediate type instructions in which a constant operand value is encoded into the ~~immediate type~~ instruction ~~itself~~ in an immediate field and in response determining that the immediate type instructions require the complex nodes.

7. (Currently amended) The method of claim 1, wherein ~~a~~ each of the complex nodes may be decomposed into a plurality of the base nodes to represent the same semantics of an instruction in the decoded program code.

8. (Currently amended) The method of claim 1, ~~wherein the program code is designed to be executed by a subject architecture, the method~~ further comprising the step of generating the complex nodes only for those features which are correspondingly configurable on the subject computing architecture.

9. (Previously presented) The method of claim 1, wherein the plurality of possible types of nodes further include polymorphic nodes.

10. (Currently amended) The method of claim 9, wherein the ~~subject~~ program code is ~~designed for execution on a subject architecture and is~~ dynamically translated into the target code for execution on ~~a~~ the target ~~architecture~~ computing system, said method further comprising:

generating the intermediate representation to include the polymorphic nodes, wherein the polymorphic nodes each contain a function pointer to a function of the target ~~architecture~~ computing system specific to a particular instruction in the ~~subject~~ program code.

11. (Currently amended) The method of claim 10, said method further comprising generating the polymorphic nodes when the features of the target ~~architecture~~ computing system would cause the semantics of a particular ~~subject~~ instruction in the program code to be lost if realized as the base nodes.

12. (Currently amended) The method of claim 10, wherein each polymorphic node is specific to a combination of a particular instruction in the ~~subject~~ program code and a function of the target ~~architecture~~ computing system.

13. (Currently amended) The method of claim 10, wherein:

said determining step further comprises identifying an instruction in ~~subject~~ the program code which corresponds an instruction on a list of polymorphic instructions to be realized as the polymorphic nodes; and

said generating step generates the polymorphic nodes only for those ~~subject~~ instructions in the program code corresponding to those on the list of polymorphic instructions~~, when a subject instruction corresponds to an instruction on the list of polymorphic instructions, when a subject instruction corresponds to an instruction on the list of polymorphic instructions~~.

14. (Previously presented) The method of claim 1, wherein the plurality of possible types of nodes further include architecture specific nodes.

15. (Currently amended) The method of claim 14, wherein the ~~subject~~ program code is ~~designed for execution on a subject architecture and is~~ dynamically translated into the target code for execution on ~~a~~ the target ~~architecture~~ computing system having a target architecture, said method further comprising:

gencrating the intermediate representation to include the architecture specific nodes which are specific to a particular combination of ~~a~~ the subject computing architecture and ~~a~~ the target architecture.

16. (Currently amended) The method of claim 15, wherein the generating step further comprises:

initially representing all of the instructions in the ~~subject~~ program code as subject architecture specific nodes, where each subject architecture specific node corresponds to a respective instruction in the ~~subject~~ program code;

determining whether an instruction in the ~~subject~~ program code is one in which to provide a target architecture specialized conversion function;

converting the subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generating the base nodes from the remaining subject architecture specific nodes which are not identified as providing ~~a~~ the target architecture specialized code generation function.

17. (Currently amended) The method of claim 16, further comprising generating ~~corresponding~~ the target code from the target architecture specific nodes, wherein the target code ~~which~~ is specialized for the target architecture.

18. (Currently amended) The method of claim 15, further comprising generating ~~corresponding~~ the target code from the base nodes, wherein the target code ~~which~~ is not specialized for the target architecture.

19. (Canceled)

20. (Currently amended) ~~A computer readable storage medium having translator software resident thereon in the form of computer readable code executable by a computer to perform the following steps during translation of subject program code to target program code:~~

~~decoding a plurality of instructions in the subject code;~~

~~generating an intermediate representation of the decoded instructions including providing a plurality of nodes in the intermediate representation as abstract representations of the expressions, calculations, and operations performed by the instructions of the subject code selected from a plurality of possible types of nodes including at least base nodes and complex nodes, wherein the base nodes represent the most basic semantics of the subject code such that the semantics of base nodes cannot be decomposed into other nodes representing more simple semantics, and wherein the~~

~~complex nodes provide a more compact representation of the semantics of complex instructions in the program code than that of base node representations; and~~

~~determining at least one type of node out of the plurality of possible types of nodes to generate in the intermediate representation for each respective instruction in the decoded subject code.~~ A computer readable storage medium having translator software resident thereon in the form of computer readable code executable by a target computer system to perform the steps of:

decoding a plurality of instructions in a program code of a subject computing architecture;

determining which type of nodes to generate in an intermediate representation for each of the decoded instructions in the program code, including determining that one or more of the decoded instructions require base nodes having basic RISC-like functionality which provides an expanded representation of semantics of the decoded instructions, and one or more of the decoded instructions require complex nodes having complex CISC-like functionality which provides a compact representation of semantics of the decoded instructions;

generating the intermediate representation of the decoded instructions using the determined types of nodes, including generating the base nodes and the complex nodes in the intermediate representation from the respective decoded instructions;

generating the target code from the intermediate representation; and

executing the target code on the target computing system.

21. (Canceled)

22. (Canceled)

23. (Currently amended) The computer readable storage medium of claim 20, wherein the base nodes are generic across a plurality of possible subject computing architectures.

24. (Canceled)

25. (Currently amended) The computer readable storage medium of claim 20, wherein the determining step includes that the program code includes ~~the complex nodes represent~~ immediate type instructions in which a constant operand value is encoded into the ~~immediate type~~ instruction

~~itself~~ in an immediate field and in response determining that the immediate type instructions require the complex nodes.

26. (Currently amended) The computer readable storage medium of claim 20, wherein ~~a~~ each of the complex nodes may be decomposed into a plurality of the base nodes to represent the same semantics of an instruction in the decoded program code.

27. (Currently amended) The computer readable storage medium of claim 20, ~~wherein the subject program code is designed to be executed by a subject architecture, the method~~ further comprising the step of generating the complex nodes only for those features which are correspondingly configurable on the subject computing architecture.

28. (Previously presented) The computer readable storage medium of claim 20, wherein the plurality of possible types of nodes further include polymorphic nodes.

29. (Currently amended) The computer readable storage medium of claim 28, wherein the ~~subject~~ program code is ~~designed for execution on a subject architecture and is~~ dynamically translated into target code for execution on ~~a~~ the target ~~architecture~~ computing system, said translator software further containing computer readable code executable by a computer to perform the following steps:

generating the intermediate representation to include the polymorphic nodes, wherein the polymorphic nodes contain a function pointer to a function of the target ~~architecture~~ computing system specific to a particular instruction in the ~~subject~~ program code.

30. (Currently amended) The computer readable storage medium of claim 29, said translator software further containing computer readable code executable by a computer to generate the polymorphic nodes when the features of the target ~~architecture~~ computing system would cause the semantics of a particular ~~subject~~ instruction in the program code to be lost if realized as the base nodes.

31. (Currently amended) The computer readable storage medium of claim 29, wherein each polymorphic node is specific to a combination of a particular instruction in the ~~subject~~ program code and a function of the target ~~architecture~~ computing system.

32. (Currently amended) The computer readable storage medium of claim 29, wherein said computer readable code executable by a computer for determining the type of nodes further:

identifies an instruction in ~~subject~~ the program code which corresponds an instruction on a list of polymorphic instructions to be realized as the polymorphic nodes; and

generates the polymorphic nodes only for those ~~subject~~ instructions in the program code corresponding to those on the list of polymorphic instructions~~, when a subject instruction corresponds to an instruction on the list of polymorphic instructions~~.

33. (Previously presented) The computer readable storage medium of claim 20, wherein the plurality of possible types of nodes further include architecture specific nodes.

34. (Currently amended) The computer readable storage medium of claim 33, wherein the ~~subject~~ program code is ~~designed for execution on a subject architecture and is~~ dynamically translated into the target code for execution on ~~a~~ the target ~~architecture~~ computing system having a target architecture, said translator software further containing computer readable code executable by a computer to perform the following steps:

generating the intermediate representation to include the architecture specific nodes which are specific to a particular combination of ~~a~~ the subject computing architecture and ~~a~~ the target architecture.

35. (Currently amended) The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to perform the following steps:

initially representing all of the instructions in the ~~subject~~ program code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the ~~subject~~ program code;

determining whether an instruction in the ~~subject~~ program code is one in which to provide a target architecture specialized conversion function;

converting the subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generating the base nodes from the remaining subject architecture specific nodes which are not identified as providing ~~a~~ the target architecture specialized code generation 14 function.

36. (Currently amended) The computer readable storage medium of claim 35, wherein said translator software ~~further~~ contains~~ing~~ computer readable code executable by a computer to generate ~~corresponding~~ the target code from the target architecture specific nodes which is specialized for the target architecture.

37. (Currently amended) The computer readable storage medium of claim 34, said translator software further containing computer readable code executable by a computer to generate ~~corresponding~~ the target code from the base nodes which is not specialized for the target architecture.

38. (Currently amended) ~~A translator apparatus for use in a target computing environment having a processor and a memory coupled to the processor for translating subject program code appropriate in a subject computing environment to produce target program code appropriate to the target computing environment, the translator apparatus comprising:~~

~~a decoding mechanism configured to decode instructions in the subject program code;~~

~~an intermediate representation generating mechanism configured to generate an intermediate representation of the decoded instructions including providing a plurality of nodes in the intermediate representation as abstract representations of the expressions, calculations, and operations performed by the instructions of the subject program code selected from a plurality of possible types of nodes including at least base nodes and complex nodes, wherein the base nodes represent the most basic semantics of the subject program code such that the semantics of the base nodes cannot be decomposed into other nodes representing more simple semantics, and wherein the~~

~~complex nodes provide a more compact representation of the semantics of complex instructions in the subject program code than that of base node representations; and~~

~~an intermediate representation type determining mechanism configured to determine which type of nodes to generate in the intermediate representation for each respective instruction in the decoded subject program code.~~ A translator apparatus for use in a target computing environment having a processor and a memory coupled to the processor for converting subject program code appropriate to a subject computing architecture to produce target code appropriate to the target computing environment, comprising:

a decoding mechanism to decode a plurality of instructions in the subject program code;

a type determining mechanism to determine which type of nodes to generate in an intermediate representation for each of the decoded instructions in the program code, including determining that one or more of the decoded instructions require base nodes having basic RISC-like functionality which provides an expanded representation of semantics of the decoded instructions, and one or more of the decoded instructions require complex nodes having complex CISC-like functionality which provides a compact representation of semantics of the decoded instructions;

an intermediate representation generating mechanism to generate the intermediate representation of the decoded instructions using the determined types of nodes, including generating the base nodes and the complex nodes in the intermediate representation from the respective decoded instructions;

a target code generating mechanism to generate the target code from the intermediate representation; and

a target code execution mechanism to execute the target code on the processor of the target computing environment.

39. (Canceled)

40. (Canceled)

41. (Currently amended) The translator apparatus of claim 38, wherein base nodes are generic across a plurality of possible subject computing architectures.

42. (Canceled)

43. (Previously presented) The translator apparatus of claim 38, wherein the complex nodes represent immediate type instructions in which a constant operand value is encoded into the immediate type instruction itself in an immediate field.

44. (Currently amended) The translator apparatus of claim 38, wherein ~~a~~ the complex nodes may be decomposed into a plurality of the base nodes to represent the same semantics of an instruction in the decoded program code.

45. (Currently amended) The translator apparatus of claim 38, wherein the program code is designed to be executed by a subject architecture, the intermediate representation generating mechanism further ~~comprising~~ comprises a complex node generating mechanism for generating the complex nodes only for those features correspondingly configurable on the subject computing architecture.

46. (Previously presented) The translator apparatus of claim 38, wherein the plurality of possible types of nodes further include polymorphic nodes.

47. (Currently amended) The translator apparatus of claim 46, wherein ~~the subject program code designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture,~~ the intermediate representation generating mechanism further comprises: ~~comprising~~:

a polymorphic node generating mechanism for generating the intermediate representation to include the polymorphic nodes,

wherein the polymorphic nodes contain a function pointer to a function of the target architecture specific to a particular instruction in the subject code.

48. (Currently amended) The translator apparatus of claim 47, said polymorphic node generating mechanism generating the polymorphic nodes when the features of the target architecture would cause the semantics of a particular ~~subject~~ instruction in the subject program code to be lost if realized as the base nodes.

49. (Currently amended) The translator apparatus of claim 47, wherein each polymorphic node is specific to a combination of a particular instruction in the subject program code and a function of the target architecture.

50. (Currently amended) The translator apparatus of claim 47, wherein said intermediate representation type determining mechanism further comprises a polymorphic identification mechanism for identifying an instruction in subject program code which corresponds an instruction on a list of polymorphic instructions to be realized as the polymorphic nodes; and

when ~~a subject~~ the instruction in the subject program code corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating mechanism generates the polymorphic nodes only for ~~these subject~~ the instructions corresponding to those on the list of polymorphic instructions.

51. (Previously presented) The translator apparatus of claim 38, wherein the plurality of possible types of nodes further include architecture specific nodes.

52. (Currently amended) The translator apparatus of claim 51, ~~wherein the subject program code is designed for execution on a subject architecture and is dynamically translated into target code for execution on a target architecture,~~ said intermediate representation generating mechanism further comprising:

an architecture specific node generating mechanism for generating the intermediate representation to include architecture specific nodes which are specific to a particular combination of a subject architecture and a target architecture.

53. (Currently amended) The translator apparatus of claim 52, the intermediate representation generating mechanism being configured to:

initially represent all of the instructions in the subject program code as subject architecture-specific nodes, where each subject architecture specific node corresponds to a respective instruction in the subject program code;

determine whether an instruction in the subject program code is one in which to provide a target architecture specialized conversion function;

convert the subject architecture specific nodes into target architecture specific nodes for those instructions determined to provide a target architecture specialized conversion function; and

generate the base nodes from the remaining subject architecture specific nodes which are not identified as providing a target architecture specialized code generation function.

54. (Currently amended) The translator apparatus of claim 52, further comprising a specialized target code generating mechanism for generating ~~corresponding~~ the target code from the target architecture specific nodes which is specialized for the target architecture.

55. (Currently amended) The translator apparatus of claim 52, further comprising a non specialized target code generating mechanism for generating ~~corresponding~~ the target code from the base nodes which is not specialized for the target architecture.

56. (Original) The translator apparatus of claim 47, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

57. (Original) The translator apparatus of claim 47, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

58. (Currently amended) The translator apparatus of claim 50, wherein when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating mechanism generates either the polymorphic nodes or the base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

59-79 (Canceled)

80. (Previously presented) The method of claim 10, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

81. (Previously presented) The method of claim 10, wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.

82. (Currently amended) The method of claim 13, wherein when a subject instruction corresponds to an instruction on the list of polymorphic instructions, said intermediate representation generating step generates either the polymorphic nodes or the base nodes for those subject instructions corresponding to those on the list of polymorphic instructions.

83. (Currently amended) The method of claim 1, wherein the translation is a dynamic binary translation from the subject program code as binary machine code of a subject instruction set architecture into the target code as binary machine code of a target instruction set architecture.

84. (Previously presented) The computer-readable storage medium of claim 29, wherein said generated polymorphic nodes specify the registers to be allocated during target code generation.

85. (Previously presented) The computer-readable storage medium of claim 29 wherein said generated polymorphic nodes are utilized in generic kernel optimizations by inferring information from the function pointer in the polymorphic node which may otherwise be indeterminable from the polymorphic node.